Mapping Wireless Communication Algorithms to a Reconfigurable Architecture

G.K. Rauwerda, G.J.M. Smit, L.F.W. van Hoesel, P.M. Heysters
University of Twente
faculty of Electrical Engineering, Mathematics & Computer Science
P.O. Box 217, 7500 AE Enschede, the Netherlands
Email: G.K.RAUWERDA@UTWENTE.NL

***Abstract*— This paper discusses the implementation of multi-standard communication systems in dynamically reconfigurable heterogeneous hardware. An overview of a wireless LAN communication system, namely HiperLAN/2, and a Bluetooth communication system will be given. Possible implementations of these systems in a reconfigurable architecture are discussed. Suggestions for future activities in the *Adaptive Wireless Networking* project are also given.**

***Index Terms*— Reconfigurable heterogeneous architecture, System on a Chip, MONTIUM, Wireless communication algorithms, Software defined radio**

I. INTRODUCTION

Traditionally, computations are either implemented in hardware or in software running on processors. More recently, however, new alternatives are introduced which mix properties of the traditional hardware and software alternatives. The class of these new alternatives is denoted as *reconfigurable computing*. This class of architectures is important because it allows the computational capacity of the machine to be highly customized to the instantaneous needs of an application while also allowing the computational capacity to be reused in time.

A key issue of systems that have to support future mobile networks is that they have to be adaptive. These systems have to adapt to changing environmental conditions (e.g. more or less users in a cell or varying noise figures due to reflections or user movements) as well as to changing user demands (QoS). Furthermore, these architectures have to be extremely efficient as these are used in battery-operated terminals and cost effective as they are used in consumer products. Although energy-efficiency is a major issue in terminals as they draw their energy from small batteries, energy consumption is also an issue in base stations from a technical (costly cooling of chips and power supplies) as well as from an environmental point of view.

In the *Adaptive Wireless Networking (AWGN)* project we aim at implementation of multi-standard communication systems in reconfigurable embedded systems. One of the tasks in the project is to map the algorithms found in these communication systems on a platform of heterogeneous reconfigurable architectures. The platform is heterogeneous in this sense that processing is performed in general purpose processors (GPPs), bit-level reconfigurable hardware or in word-level reconfigurable hardware. It is expected that performance and power gains will be achieved by applying dynamically reconfigurable heterogeneous architectures [1].

In this paper we will give an introduction to the *Adaptive Wireless Networking* project. Research, related to our project, will be briefly described in Section II. An overview of a particular reconfigurable heterogeneous architecture will be given in Section III. In Section IV some basics about the HiperLAN/2 and Bluetooth communication system will be discussed. These systems are currently under investigation in our project. Once the basic properties of the communication systems are known, we will give possible implementations of these systems in reconfigurable hardware in Section V. The obtained results of these implementations are shown in Section VI. Some conclusions will be presented in Section VII and recommendations for future work will be presented in Section VIII.

II. RELATED WORK

Recently, there have been several announcements of heterogeneous reconfigurable architectures on a single chip [2], [3]. For example, Xilinx delivers a combination of Virtex II FPGA technology and one or more PowerPC 405 – on a single chip – the Virtex-II Pro. But conventional reconfigurable processors are bit-level reconfigurable and are far from energy-efficient. At the University of Twente in the *Chameleon* project [4] the first steps have been made to define an energy-efficient heterogeneous reconfigurable System-on-Chip architecture.

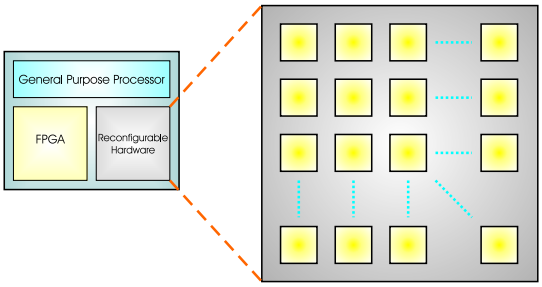

Fig. 1. CHAMELEON heterogeneous SoC architecture

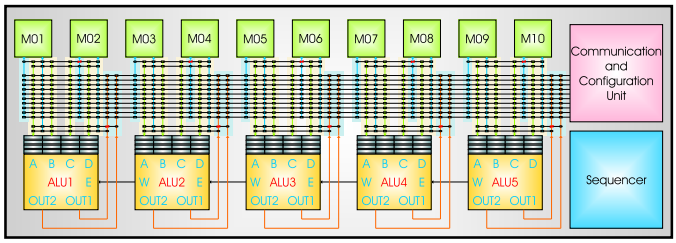

Fig. 2. MONTIUM processor tile

Also much work has been done on software defined radios in the SDR forum context [5]. However, this forum mainly focuses on general-purpose processors and they do not concentrate on reconfigurable platforms and energy-efficiency. *Pleiades* at the University of California, Berkeley, is exploring reconfiguration of coarser-grain application-specific building blocks with an emphasis on low-power computations [6]. Chameleon Systems created one of the first practical commercial implementation of coarse-grain reconfigurable technology for data intensive Internet, DSP and other high performance telecommunication applications [7].

III. RECONFIGURABLE HETEROGENEOUS ARCHITECTURE

In the *Chameleon* project [4] at the University of Twente a dynamically reconfigurable heterogeneous System-on-a-Chip (SoC) is being defined. The SoC contains a general purpose processor, a fine-grained reconfigurable part (consisting of FPGA tiles) and a coarse-grained reconfigurable part. The latter comprises several MONTIUM processor tiles. The algorithm domain of the MONTIUM comprises 16-bit digital signal processing (DSP) algorithms that contain multiply accumulate (MAC) operations. A MONTIUM-tile is designed to execute highly regular computational intensive DSP kernels. The irregular parts of the algorithms run on the general purpose processor. The SoC yields a combination of performance, flexibility and energy-efficiency.

A. *Target architecture:* MONTIUM

In this section we give a brief overview of the MONTIUM architecture, because in this paper this architecture is used for mapping DSP algorithms of wireless communication systems. More details can be found in [4], [8], [9], [10]. Fig. 2 depicts a single MONTIUM processor tile. The hardware organisation within a tile is very regular and resembles a very long instruction word (VLIW) architecture. The five identical arithmetic and logic units (ALU1$\cdots$ALU5) in a tile can exploit spatial concurrency to enhance performance. This parallelism demands a very high memory bandwidth, which is obtained by having 10 local memories (M01$\cdots$M10) in parallel. The small local memories are also motivated by the locality of reference principle. The ALU input registers provide an even more local level of storage. Locality of reference is one of the guiding principles applied to obtain energy-efficiency in the MONTIUM. A vertical segment that contains one ALU together with its associated input register files, a part of the interconnect and two local memories is called a processing part (PP). The five processing parts together are called the processing part array (PPA). A relatively simple sequencer controls the entire PPA. The communication and configuration unit (CCU) implements the interface with the world outside the tile. The MONTIUM has a datapath width of 16-bits and supports both integer and fixed-point arithmetic. Each local SRAM is 16-bit wide and has a depth of 512 positions, which adds up to a storage capacity of 8 Kbit per local memory. A memory has only a single address port that is used for both reading and writing. A reconfigurable address generation unit (AGU) accompanies each memory. The AGU contains an address register that can be modified using base and modify registers.

It is also possible to use the memory as a lookup table for complicated functions that cannot be calculated using an ALU, such as sinus or division (with one constant). A memory can be used for both integer and fixed-point lookups. The interconnect provides flexible routing within a tile. The configuration of the interconnect can change every clock cycle. There are ten busses that are used for inter-PPA communication. Note that the span of these busses is only the PPA within a single tile. The CCU is also connected to the global busses. The CCU uses the global busses to access the local memories and to handle data in streaming algorithms. Communication within a PP uses the more energy-efficient local busses. A single ALU has four 16-bit inputs. Each input has a private input register file that can store up to four operands. The input register file cannot be bypassed, i.e., an operand is always read

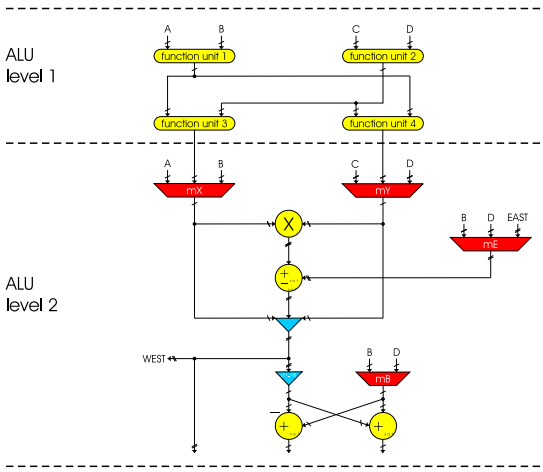

Fig. 3. MONTIUM ALU

from an input register. Input registers can be written by various sources via a flexible interconnect. An ALU has two 16-bit outputs, which are connected to the interconnect. The ALU is entirely combinatorial and consequentially there are no pipeline registers within the ALU. The diagram of the MONTIUM ALU in Fig. 3 identifies two different levels in the ALU. Level 1 contains four function units. A function unit implements the general arithmetic and logic operations that are available in languages like C (except multiplication and division). Level 2 contains the MAC unit and is optimized for algorithms such as FFT and FIR. Levels can be bypassed (in software) when they are not needed.

Neighbouring ALUs can also communicate directly on level 2. The West-output of an ALU connects to the East-input of the ALU neighbouring on the left (the West-output of the leftmost ALU is not connected and the East-input of the rightmost ALU is always zero). The 32-bit wide East-West connection makes it possible to accumulate the MAC result of the right neighbour to the multiplier result (note that this is also a MAC operation). This is particularly useful when performing a complex multiplication, or when adding up a large amount of numbers (up to 20 in one clock cycle). The East-West connection does not introduce a clock delay, as it is not registered.

IV. COMMUNICATION SYSTEMS

The research presented in this paper is focused on multi-standard communication systems. By studying the state-of-the-art in Software Defined Radio (SDR) for Bluetooth, HiperLAN/2 and UMTS, we will define a set of typical DSP algorithms used in wireless communications. In this paper we present the initial results of mapping a set of characteristic algorithms on our dynamically reconfigurable heterogeneous platform. Currently, the physical layers of two communication systems are subject of our research: a rather complex wireless LAN system, the HiperLAN/2 standard, and a less complex communication system, the Bluetooth standard. These two communication standards have been selected, because they are already part of ongoing research at the University of Twente [11] and they are different enough to give an indication whether our approach is feasible or not.

A. *HiperLAN/2 receiver*

The task of the physical layer in HiperLAN/2 is to modulate bits that originate from the data link control layer on the transmitter side and to demodulate them on the receiver side. The transmission format of the physical layer is a burst, which consists of a preamble and a data part.

Orthogonal frequency division multiplexing (OFDM), which is a special kind of multicarrier modulation, has been used in HiperLAN/2. The modulation technique divides the high data rate information in several parallel bit streams and each of these bit streams modulates a separate subcarrier. 52 subcarriers are being transmitted in parallel per radio channel, which occupies a bandwidth of 20 MHz. However, 4 of these subcarriers are used to transmit pilot tones [12].

One OFDM symbol is represented by 80 complex input samples (in the time domain), so once per 80 samples the receiver has enough information to demodulate the received samples and output the resulting bits. The sample rate of the input signal is 20 MHz, so the duration of an OFDM symbol is 4 μs.

The receiver not only has to convert the received signal to data bits by performing the inverse of the transmitter, but it also has to try to inverse distortions caused by the radio channel. The receiver can roughly be divided into two parts, a time domain part and a frequency domain part.

In the first stage of the receiver, signal functions will be time domain functions. In the second stage of the receiver, signal functions will be frequency domain functions. The location of the functions in the receiver architecture, shown in Fig. 4, is based upon a trade-off between the necessary resolution that must be reached for a certain correction and the solution with the minimum number of operations. One also tries to keep the corrections independent of each other by deciding the execution order of the functions [13].

We will describe the most important functions (most computational intensive) that are being subjected to further research, in order to map the functionality on the MONTIUM architecture. We only consider the receiver part of the HiperLAN/2 physical layer, although the transmitter part can be described in a similar way.

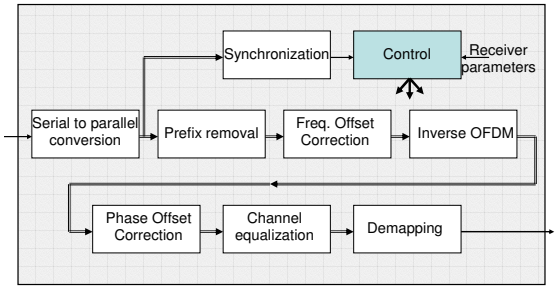

Fig. 4. Demodulator part of the HiperLAN/2 receiver

1) Prefix removal: The cyclic prefix of a data OFDM symbol contains a copy of the last 16 samples of the useful part of that symbol. Hence this redundant information can be used to find the beginning of the symbol. Therefore the correlation is calculated between 16 samples and 16 samples received 64 samples earlier.

In order to avoid a waste of effort to calculate this correlation for each incoming sample, an estimate is made for the location of the symbol start. During the search for the symbol start, the length of the cyclic buffer is set to 96 samples. This gives the symbol window the opportunity to move eight samples in both directions, since a symbol is only 80 samples long. Because the symbol window is 16 samples larger than the actual symbol, the outcome of the Prefix removal function has to be determined sixteen times, each time advancing one sample [13].

2) Inverse OFDM: 64 time domain samples represent the useful data part of the OFDM symbol that has to be demodulated. Before demodulation can take place, the subcarrier values must be retrieved from the useful data part. This can be done by applying a fast Fourier transform (FFT) to the vector containing the 64 samples. The FFT efficiently implements a discrete Fourier transform (DFT), given in Eq. 1.

$$\hat{f}_n[x] = \sum_{m=0}^{N-1} \tilde{s}_n[m] e^{-j2\pi \frac{xm}{N}} \qquad (1)$$

with $x = 0, ..., 63$ and $N = 64$. $\tilde{s}_n$ denotes the vector of input samples in the time domain. $\hat{f}_n$ denotes the vector of samples in the frequency domain. From this vector the 52 subcarrier values can be extracted.

3) De-mapping: In HiperLAN/2 there are four mapping techniques available: BPSK, QPSK, 16-QAM and 64-QAM. Each of these techniques has a different number of bits per complex symbol. By way of a lookup table the output bits are determined. In the lookup table, all possible subcarrier values for a certain mapping scheme are defined. The most likely symbol that was transmitted is the symbol to which the Euclidian distance (Eq. 2) in the lookup table is smallest.

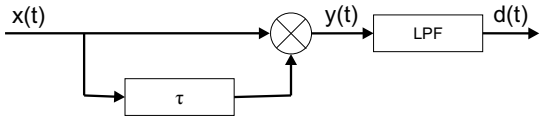

Fig. 5. Block diagram of the FM-discriminator

$$|a - b| \equiv \sqrt{(\Re(a) - \Re(b))^2 + (\Im(a) - \Im(b))^2} \qquad (2)$$

B. Bluetooth receiver

The task of the physical layer in Bluetooth is to modulate bits that origin from the data layer on the transmitter side and to demodulate them on the receiver side, and vice versa. The Bluetooth system uses packet-based transmission: the information stream is fragmented into packets. In each slot, only a single packet can be sent. All packets have the same format, starting with an access code, followed by a packet header, and ending with the user payload. Single slot, 3-slot and 5-slots packets have been defined.

The channel is a hopping channel with a nominal hop dwell time of 625 μs that corresponds to a single slot. To simplify the implementation, full-duplex communication is achieved by applying time-division duplexing (TDD).

In the Industrial, Scientific and Medical (ISM) band, the signal bandwidth of the Bluetooth system is limited to 1 MHz. For robustness, a binary modulation scheme was chosen. With the mentioned bandwidth restriction, the data rates are limited to about 1 Mbps. Bluetooth uses Gaussian-shaped frequency shift keying (GFSK) modulation with a nominal modulation index of $k = 0.32$. Logical ones are sent as positive frequency deviations, logical zeros as negative frequency deviations [14].

In the receiver part, the transmitted radio signal is converted back into a binary NRZ signal. The radio signal, which is received, conveys all its information in the frequency deviation of the signal.

1) FM discriminator: One possible way to demodulate a FM signal is by way of FM-to-AM conversion, which is also called a FM-discriminator. The FM-discriminator allows the implementation of low-cost radio units, which is essential for Bluetooth systems. In the FM-discriminator, which is shown in Fig. 5, the received signal is multiplied with its delayed version. The

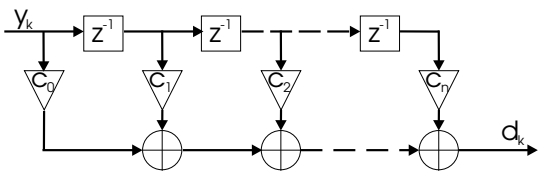

Fig. 6. Finite Impulse Response filter

signal has to be delayed $\tau = \frac{1}{4f_c}$ seconds [15]. When the center frequency, f_c, of the intermediate Bluetooth signal is 2.5 MHz, one has to set the delay, τ, to 100 ns. The sample rate of the Bluetooth signal in the SDR testbed is equivalent to 10 MSPS. Consequently, the signal has to be delayed by 1 sample time.

2) Low Pass Filter: After FM-to-AM conversion, the signal is passed through a Low Pass Filter. The FIR filter is applied in order to pass the slowly varying AM signal with a frequency of 1 MHz and to block all high frequencies that occur due to multiplication. Actually the FIR filter, as depicted in Fig. 6, operates at a sampling rate of 10 MSPS.

3) Threshold detector: Finally, the consecutive bits are detected by a threshold detector. The threshold detector is not shown in Fig. 5. In the threshold detector decisions about the received bit value are taken. For the threshold detector it is sufficient to operate at a sample rate of 1 MSPS, since the bit rate of the Bluetooth communication system is 1 Mbps.

V. Implementation

For both communication systems we will analyze the feasibility of implementing the systems in the target architecture. We consider the MONTIUM architecture and assume that the clock frequency of the MONTIUM-tiles is 100 MHz. So one clock cycle has a duration of 10 ns.

A. HiperLAN/2 receiver

1) Prefix removal: In the prefix removal function (and also the synchronization function) the preambles are detected with matched filters. In Tab. I the sizes of the matched filters that are applied in the different functions are shown[1].

Because the window is 16 samples larger than the actual symbol, the outcome of the matched filter during prefix-detection has to be determined 16 times, each time advancing one sample. A peak in the consecutive outputs of the matched filter indicates the start of the OFDM symbol. During the detection of the prefix a cyclic buffer of length 96 samples is used. After the prefix is detected, the useful part of the OFDM symbol is copied to a cyclic buffer of length 64.

TABLE I
SIZES OF THE MATCHED FILTERS THAT ARE APPLIED IN DIFFERENT FUNCTIONS

Detection of	Size of matched filter
Preamble A	16 complex samples
Preamble B	16 complex samples
Preamble C	32 complex samples
Prefix	16 complex samples

In Eq. 3 the cross-correlation between two complex vectors is defined. The output from a matched filter with N samples and two complex inputs x and y is

$$output[l] = \frac{1}{N}\sum_{i=1}^{N} x[i]y[l+i]^* \qquad (3)$$

where y^* denotes the complex conjugate of y.

During matched-filtering the cross-correlation is determined in order to detect preambles and prefixes. One possible way to implement the cross-correlation function in the MONTIUM-tile is a separate approach of calculating the real-part and the imaginary-part of the correlation.

With 4 ALUs one can determine the real and imaginary part of the complex cross-correlation. In order to result in one valuable output, one has to determine the absolute value of the complex cross-correlation and also a scaling factor $1/N$ has to be applied. Since square root calculations are computationally complex, we will determine the squared absolute value of the cross-correlation instead of the absolute value. Hence the square root calculation is avoided. Scaling with $1/N$ is also neglected, since it only scales the output of the cross-correlation calculation[2]. However, the shape of the correlation function will remain the same, so peaks can still be detected.

Considering the complex cross-correlation one can conclude that 4 ALUs are utilized in order to perform a cross-correlation calculation of one complex sample. When the complex cross-correlation of N samples has to be calculated, this calculation will spend $N + 1$ clock cycles. The real part and the imaginary part of the sum can be calculated in parallel using 4 ALUs. When the entire sum for all N samples is calculated, one has to change the configuration of the ALUs in order to calculate the squared absolute value of the complex correlation. This operation will absorb one extra clock cycle. In total 2 differ-

[1] In HiperLAN/2 3 preambles and one prefix are used for synchronization.

[2] When N = power of two, a shift operation can be applied to perform scaling by $1/N$.

ent ALU configurations are used to perform the complex correlation. A summary of the requirements for complex cross-correlation is given in Tab. II.

TABLE II
REQUIREMENTS FOR A CORRELATION OF N COMPLEX SAMPLES ON THE MONTIUM ARCHITECTURE

# clock cycles	$N+1$
# ALU configurations	2
# tiles	1

2) Inverse OFDM: The inverse orthogonal frequency division multiplexing in HiperLAN/2 is just a 64-FFT operation. This operation is performed by radix-2 butterflies and consumes $64\log_2(64)$ complex multiplications. On the MONTIUM architecture the 64-FFT can be performed in $\left(\frac{64}{2}+1\right)\log_2(64)$ clock cycles [8].

$\log_2(64)+4$ different memory configurations are used to perform the 64-FFT on the MONTIUM and 8 different crossbar configurations have to be used. There is only 1 ALU configuration required, since for each operation the same radix-2 structure (butterfly) is used.

The FFT algorithm can be performed with 4 ALUs and 10 memory banks, which corresponds to one MONTIUM-tile, as given in Tab. III.

TABLE III
REQUIREMENTS FOR 64-FFT ON THE MONTIUM ARCHITECTURE

# clock cycles	198
# crossbar configurations	8
# memory configurations	10
# ALU configurations	1
# tiles	1

3) De-mapping: The de-mapping in the model is implemented as hard decision de-mapping. In HiperLAN/2 there are four modulation schemes available: BPSK, QPSK, 16-QAM and 64-QAM. The hard output bits are determined by comparing the soft output values with values in a lookup table. In Tab. IV the size of the lookup table for the different modulation schemes is given. The most likely symbol that was transmitted is the symbol to which the Euclidean distance in the lookup table is smallest.

While one first determines in which quadrant the received soft-value is situated, one has to determine the Euclidean distance in that corresponding quadrant 1, 1, 4 or 16 times for BPSK, QPSK, 16-QAM or 64-QAM, respectively. From these Euclidean distances one has to determine the minimum value (Tab. V). The hard output bits

TABLE IV
SIZE OF INFORMATION STORED IN LOOKUP TABLES

Information in lookup table	**Entries in lookup table**
Carrier values BPSK	2 complex numbers
Carrier values QPSK	4 complex numbers
Carrier values 16-QAM	16 complex numbers
Carrier values 64-QAM	64 complex numbers

are equal to the index of the minimum value in the lookup table (when the lookup table is well initialized).

TABLE V
COMPUTATIONAL REQUIREMENTS FOR DE-MAPPING ONE COMPLEX VALUE IN HIPERLAN/2

Modulation	**# Euclidean distance calculations**	**# minimum calculations**
BPSK	1	1
QPSK	1	1
16-QAM	4	4
64-QAM	16	16

Calculating the Euclidean distance requires a square root calculation, as seen in Eq. 2. Since the Euclidean distance is only calculated for determining the minimum Euclidean distance, it is also allowed to determine the minimum value of the squared Euclidean distance. In this way the square root calculation is overruled.

Using 2 ALUs one can calculate the squared Euclidean distance between the known complex value and the soft output of the FFT function. When these distances are determined, one has to determine a minimum distance, which can be performed with 1 ALU that is comparing all distances in a certain amount of clock cycles. The memory address of the corresponding soft-value corresponds to the hard output bits.

During one clock cycle one can perform two Euclidean distance calculations using one tile on the MONTIUM-architecture. Within one MONTIUM-tile one can determine two squared Euclidean distances in parallel using 4 ALUs. The fifth ALU in the MONTIUM-tile can be used to determine the minimum of the calculated distances. First the minimum from the outputs of ALU-1 and ALU-3 is determined. Hereafter this obtained minimum is compared to the local minimum, which is obtained in foregoing clock cycles. Using the mapping, given above, one can perform the searching for minimum Euclidean distance in 1, 1, 2 or 8 clock cycles for BPSK, QPSK, 16-QAM or 64-QAM, respectively (see Tab. VI). Since each OFDM symbol consists of 48 complex values, the computation

time of the entire OFDM symbol is 48 times larger than the numbers given in Tab. VI.

TABLE VI
COMPUTATIONAL REQUIREMENTS FOR DE-MAPPING ONE COMPLEX VALUE IN HIPERLAN/2

Modulation	# clock cycles
BPSK	1
QPSK	1
16-QAM	2
64-QAM	8

B. Bluetooth receiver

1) FM-discriminator: In the FM-discriminator (Fig. 5) the incoming FM signal is multiplied with its delayed version. Since the incoming signal is sampled at a frequency of 10 MHz, the signal only has to be delayed with one sample time under the assumption that the intermediate frequency of the incoming signal is 2.5 MHz.

In one MONTIUM-tile one can perform 5 multiplications in parallel. An advantage of the Bluetooth FM signals is that all sample values represent real numbers. Consequently, 5 real multiplications can be calculated in one clock-cycle.

2) FIR filter: The processing time of the FM-discriminator depends on the amount of samples that has to be processed at the initialization phase; the FIR filter has to be initialized with an amount of samples that is equal to the number of taps.

Suppose that a FIR filter with 10 taps is employed, so 10 sample values have to be generated by the FM-discriminator, before the FIR filter is initialized. Since the incoming FM modulated signal has to be delayed with one sample time, $10 + 1$ samples have to be loaded in the local memory before the signal processing can start.

The length of the FIR filter, used in the FM-discriminator, is not variable. However, in a MONTIUM-tile one can perform a FIR filter with a variable amount of taps up to a maximum of 2560 taps. In all ALUs multiply and accumulate operations can be performed. Utilizing the EAST-WEST interconnect between the ALUs gains 1 clock cycle while performing FIR filtering. When there are no EAST-WEST interconnects applied, each ALU has to store its temporary result. When the multiply and (partially) accumulate process is finished, the temporary results of all ALUs have to be collected and summed. This addition of the temporary accumulated results requires one extra clock cycle. Depending on the implementation without or with EAST-WEST interconnect, FIR filtering with N coefficients can be performed in $\frac{N}{5} + 2$ or $\frac{N}{5} + 1$ clock cycles, respectively. Not using the EAST-WEST connection has a positive influence on the maximum clock frequency of the MONTIUM-tile.

3) Threshold detector: In the threshold detector one has to decide whether a '0' or '1' is received. The output of the FIR filter depends on the frequency deviation in the received Bluetooth signal. The signal at the output of the FIR filter varies between -1 and $+1$. This signal has to be translated into 'received' bits, by applying these rules:

if $x > 0$ then bit:=1
if $x \leq 0$ then bit:=0

A decision of a bit value is performed using one ALU and takes only one clock cycle. The actual output bit is determined by the status information bit of the function unit.

VI. RESULTS

A. HiperLAN/2

Mapping the HiperLAN/2 receiver algorithms on the MONTIUM architecture requires knowledge of the typical HiperLAN/2 symbol durations. In section V we have discovered the typical computational requirements of these algorithms. Furthermore we discovered that all algorithms can be mapped on one tile. In Tab. VII we summarize the time consumption of all the algorithms, while they are performed on a single tile and the number of ALU configurations that are utilized.

One major property of signaling in HiperLAN/2 physical layer is the timing aspect of OFDM symbols. All operations in the physical layer are performed on these OFDM symbols. An OFDM symbol has a fixed length of 80 samples. Hence, at a sample rate of 20 MHz the duration corresponds to 4 μs. One should assure that each 4 μs a new OFDM symbol can be processed.

The complete receiver processing would require about 11 μs (at maximum) excluding the synchronization and control processing (see Tab. VII). Hence, one OFDM symbol can not completely be processed by all receiver algorithms within its duration of 4 μs. So scheduling of tasks on different tiles has to be performed. From the table can be seen that only 10 ALU configurations are required for performing all receiver algorithms. Because of timing constraints, while each 4 μs an OFDM symbol has to be processed, 2 or 3 MONTIUM-tiles, running a clock frequency of 100 MHz, are required in order to perform all receiver processing. The number of required tiles depends on the modulation type used in the communication system; For BPSK modulation only 2 tiles are required, while

TABLE VII

COMPUTATIONAL REQUIREMENTS OF HIPERLAN/2 RECEIVER ALGORITHMS MAPPED ON THE MONTIUM ARCHITECTURE WHILE PROCESSING 1 OFDM SYMBOL

Functional block	**# clock cycles**	**# ALU configurations**	**Computation time @ 100 MHz [μs]**
Prefix removal	272	2	2.72
Frequency offset correction	64	2	0.64
Inverse OFDM	198	1	1.98
Phase offset correction	128	2	1.28
Channel equalisation	64	2	0.64
De-mapping	48 – 384	1	0.48 – 3.84
Total			*7.74 – 11.1*

64-QAM modulation requires 3 MONTIUM-tiles. Different tasks have to be scheduled over these tiles. One extra tiles may have to be used in order to perform the synchronization and control part of the receiver.

B. Bluetooth

Mapping the Bluetooth receiver algorithms on the MONTIUM architecture requires knowledge about the typical Bluetooth symbol timing. We have seen that there are single slot, 3-slot and 5-slot packets in the standard, which have a duration of 625 μs, 1.875 ms and 3.125 ms, respectively. Actually the slots are not completely occupied by the packets. A single slot packet has a length of 366 bits at maximum, and therefore has a duration of 366 μs. The length of a 3-slot packet is at most 1626 bits, which corresponds to 1.626 ms, and for a 5-slot packet the length is 2870 bits at maximum, which is equal to a duration of 2.87 ms.

From these values of the packet lengths, one can conclude that it should be useful to perform the receiver functions in 'streaming' mode. This means that the processing of the receiver functions starts immediately, and does not wait till a complete packet has been loaded into the local memory. In Bluetooth the data rates are limited to about 1 Mbps, so at most every 1 μs a bit has to be processed by the Bluetooth receiver.

In Tab. VIII we summarize the time consumption of all the algorithms, while they are performed on a single tile and the number of ALU configurations that are utilized. The processing delays in the table are given while processing is done for a single sample. During initialization of the receiver one has to load $N + 1$ samples in the local memory before the processing can start. This initialization phase will consume $ceil\{\frac{N}{5}\} + 1$ clock cycles extra (the ceiling value function, $ceil\{x\}$, computes the smallest integer not less than x).

VII. CONCLUSION

We have analyzed the feasibility of implementing communication systems in reconfigurable hardware. In the *Adaptive Wireless Networking (AWGN)* project we will utilize a dynamically reconfigurable heterogeneous architecture for implementation of multi-standard communication systems. The architecture is heterogeneous is the sense that digital signal processing is performed in general purpose processors, bit-level reconfigurable parts or word-level reconfigurable parts.

The physical layer of the HiperLAN/2 system is discussed and the functionality of an HiperLAN/2 receiver is described using a Software Defined Radio (SDR) view. For important functions in the receiver we described the characteristics that are important while mapping onto reconfigurable hardware. During the mapping of the functions, we have used our MONTIUM architecture. In order to perform all receiver processing (excluding the synchronization and control processing part), 2 or 3 MONTIUM-tiles are required (depending on modulation type), when we assume the tiles to run at a clock frequency of 100 MHz. We estimated the processing delay in the receiver of one OFDM symbol to be about 11 μs when 64-QAM modulation is applied. Furthermore, the processing delay using BPSK modulation will be about 8 μs. Hence, we have an adaptable HiperLAN/2 receiver, which can be implemented in 2 or 3 MONTIUM-tiles depending on modulation scenario.

The functionality of the Bluetooth receiver appeared to have a fairly simple signal processing part, which can be implemented in the MONTIUM architecture quite well. The computation delays of the different receiver parts seem to be a few clock cycles. The processing while receiving one bit takes about 130 ns, when a FIR filter with 50 coefficients is applied and the clock frequency of the MONTIUM-tile is 100 MHz.

We showed the flexibility of the MONTIUM architec-

TABLE VIII

COMPUTATIONAL REQUIREMENTS OF BLUETOOTH RECEIVER ALGORITHMS MAPPED ON THE MONTIUM ARCHITECTURE WHILE PROCESSING 1 BIT

Functional block	**# clock cycles**	**# ALU configurations**	**Computation time @ 100 MHz [ns]**
FM-discriminator	1	1	10
N-taps FIR filter	$ceil\{\frac{N}{5}\}+2$	2	$ceil\{\frac{N}{5}\}\times 10+20$
	$ceil\{\frac{N}{5}\}+1$	2	$ceil\{\frac{N}{5}\}\times 10+10$
Threshold detector	1	1	10

ture. An HiperLAN/2 receiver was implemented, which is adaptable depending on the used modulation scheme, as well as a flexible Bluetooth receiver on the same architecture.

VIII. FUTURE WORK

While suggesting implementations of the Bluetooth receiver, we have not considered the channel selection functionality in the physical layer. The channel selection mechanism is very dynamic, because the subcarrier that contains the information is changing at most every 625 μs, and therefore it would show a perfect example of adaptivity and reconfigurability in reconfigurable hardware. The channel selection mechanism should be studied in detail and implementations should be suggested in order to extend the dynamically reconfigurable Bluetooth receiver.

The set of multi-standard algorithms is rather incomplete. In further research, more adaptive algorithms should be considered. An interesting case will be systems that adapt to changing environmental conditions as well as to changing user demands (QoS). A control system that is able to adapt the WCDMA receiver in order to minimize the energy consumption, while satisfying the quality constraints at run-time is presented in [16]. These adaptations should be done at run-time to deal with the continuously changing external environment.

Power consumption is another important issue, since energy-efficiency is a major issue in terminals and in base stations. At the moment, the energy consumption of the MONTIUM architecture is estimated at 5 mW per tile in .12 technology with a size of 2.7 mm^2 per tile. Comparison of the algorithm mapping as compared to other types of architectures should be done in terms of performance and power consumption.

Finally, scheduling of tasks on the reconfigurable heterogeneous architecture can be implemented in different ways: Multiple tiles can be performing different functions instantaneously, while the data in the tile is changing dynamically or the data stays in one tile, while the tile is reconfigured dynamically. Simulations have to show the advantages and disadvantages of these scheduling strategies.

ACKNOWLEDGEMENT

We thank our colleagues of the Signals & Systems group, Fokke Hoeksema and Roel Schiphorst, for their contribution to our research.

This research is supported by the Freeband Knowledge Impulse programme, a joint initiative of the Dutch Ministry of Economic Affairs, knowledge institutions and industry.

REFERENCES

[1] Jan Rabaey, *Silicon Architectures for Wireless Systems*, Tutorial Hotchips symposium, August 2001

[2] Xilinx, *Virtex-II Pro Platform FPGAs*, http://www.xilinx.com/virtex2pro/

[3] Altera, *Excalibur Embedded Processor Solutions*, http://www.altera.com/products/devices/excalibur/exc-index.html

[4] Chameleon project - *Reconfigurable computing in hand-held multimedia computers*, http://chameleon.ctit.utwente.nl

[5] SDR Forum, http://www.sdrforum.org

[6] Pleiades project - *Ultra-Low-Power Reconfigurable Computing*, http://bwrc.eecs.berkeley.edu/Research/Configurable_Architectures/

[7] Chameleon Systems, *Reconfigurable Communications Processor*, http://www.chameleonsystems.com

[8] Paul M. Heysters, Henri Bouma, Jaap Smit, Gerard J.M. Smit, Paul J.M. Havinga, *A Reconfigurable Function Array Architecture for 3G and 4G Wireless Terminals*, Proceedings of 2002 World Wireless Congress, pages 399-404, San Francisco, USA, May 2002

[9] P.M. Heysters, G.J.M. Smit, E. Molenkamp, *A Flexible and Energy-Efficient Coarse-Grained Reconfigurable Architecture for Mobile Systems*, **accepted for** Journal of Supercomputing, Kluwer, 2003

[10] P.M. Heysters, G.J.M. Smit, E. Molenkamp, *Montium - Balancing between Energy-Efficiency, Flexibility and Performance*, **accepted for** The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'03), 2003

[11] Software-Defined-Radio project - *A Bluetooth-HiperLAN/2 SDR receiver*, http://nt5.el.utwente.nl/sdr/

[12] ETSI, *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer*, ETSI TS 101 475 V1.2.2 (2001-02), 2001

[13] L.F.W. van Hoesel, *Design and implementation of a software defined HiperLAN/2 physical layer model for simulation purposes*, Master of Science Thesis, University of Twente, Enschede, August 2002

[14] Bluetooth SIG, *Specification of the Bluetooth System - Core*, Technical Specification Version 1.1, 22 February 2001

[15] Roel Schiphorst, Fokke Hoeksema, Kees Slump, *Bluetooth demodulation algorithms and their performance*, 2nd Karlsruhe Workshop on Software Radios, pages 99-106, March 2002

[16] Lodewijk T. Smit, Gerard J.M. Smit, Paul J.M. Havinga, Johann L. Hurink and Hajo Broersma, *Influences of RAKE Receiver/Turbo Decoder Parameters on Energy Consumption and Quality*, Proceedings of 2002 World Wireless Congress, pages 175-180, San Francisco, USA, May 2002